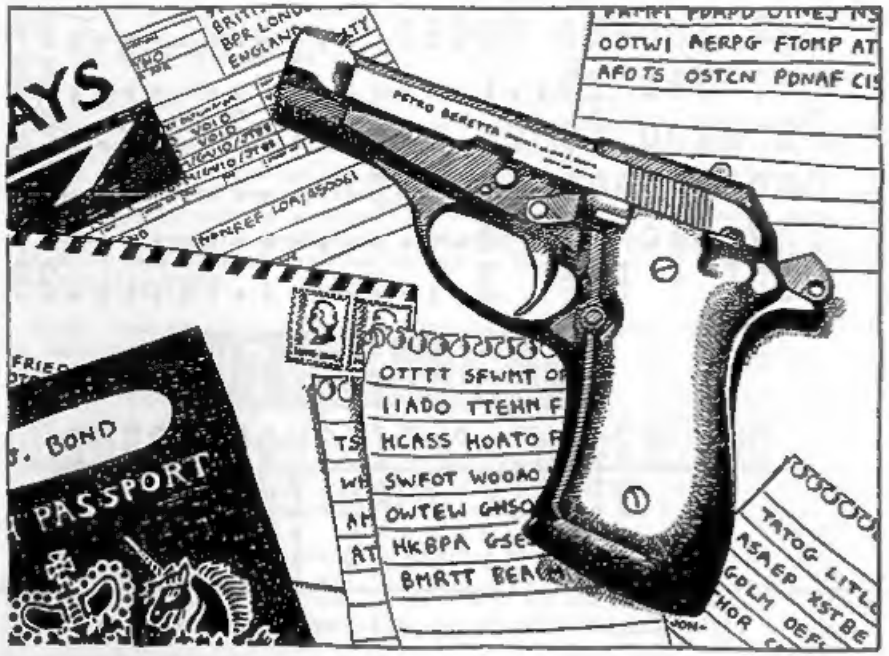
Vol 2 - No 4. November 1988.

FORMAT

THE MONTHLY MAGAZINE FOR SPECTRUM, DISCiPLE & PLUS D USERS



CIPHER MACHINE

A LICENCE TO SPY...

Vol 2 No 4. # CONTENTS Nov 1988.

NEXT MONTH

OUR SECOND CHRISTMAS ISSUE
BIGGER THAN EVER
DONT MISS IT

FORMAT is published by INDUG. 34 Bourton Road, Gloucester, GL4 0LE, England. Telephone 0452-412572. DISCIPLE and PLUS D are trade marks of MILES GORDON TECHNOLOGY, Lakeside Technology Park, Phoenix Way, Swansea, South Wales, SA7 9EH. Telephone 0792-791100. The DISCIPLE is marketed by Rockfort Products, 81 Church Road, Hendon, London, NW4 4DP.

Printed by O.S.LITHO. Gloucester. Tels- (0452) 23198

The Editor Speaks

FORMAT is going through a period of change at the moment. I hope that what comes out the other end will be to everyones liking. First, we are widening the scope of the magazine to include far more general Spectrum articles. Disc users need not despair, there will still be plenty about the DISCiPLE and PLUS D. But FORMAT is now the only serious magazine produced (as far as I know) and I feel its time we expanded our readership to non disc users. An advert will be appearing in one of the glossies over the next month and it is hoped that subscribers will rise in number very quickly. This will enable FORMAT to grow in size as well as scope.

If you have friends who are Spectrum owners, why not do them a favour and get them to subscribe to FORMAT. To give you an incentive (if doing you friend a favour is not enough) we will extend your subscription by 2 months for each new person who joins. Just send me a letter, giving the name and address of your friend, and we will send them a subscription form. As soon as we receive this back your membership will be adjusted. Just think, 5 new subscribers introduced, and your next years FORMATs are totally FREE.

From now on there's a new Special Offer from MGT. In the past we have sent out a Special Offer leaflet to new members, but this has always been a small sub-set of the items MGT have on their current catalogue. We now think we have found an easier way, from now on MGT are offering an across the board - 5% discount - to INDUG members, yes 5% discount on their entire range of Hardware, Software and Accessories. In addition it will no longer be necessary to send a chegue through FORMAT, just send it direct to Swansea or give MGT a ring on 0792-791100.

I spent a very interesting week in Swansea at the beginning of the month having talks with Bruce Gordon and Alan Miles about SAM. Details of the screen modes will be found in this issue and I have a lot more in store for you in the near future.

Finally this month I have a complaint to make. We have a letters page in FORMAT. This is were YOU can have YOUR say on any computer related subject, anything of interest to other users. But each month less and less comes in. What are you doing out there, just sitting down waiting for FORMAT each month? Several people have asked for a larger letters page but I can't print letters I haven't received. So please, get writing.

See you next month.

Bob Brenchley. Editor.

NEWS ON 4

NEW DISC MANAGER FROM BETTERBYTES

BETTERBYTES Software of Newcastle have now released a new version of their highly successful DISC MANAGER system. The new version contains many completely new features together with a major rewrite of several of the existing ones. DISC MANAGER consists of 6 programs (plus 6 supplementary), totalling over 250k. For an extra special price to FORMAT readers see BETTERBYTES advert in this issue.

Dave Hood, the author of the program, believes the new version is faster, more advanced and more user friendly than previous versions. "A new machine code sort and search has really improved the speed of the system" said Dave.

At the same time BETTERBYTES are also releasing a new program called DISC ORGANISER. This performs several disc utility functions including RECOVER erased programs and ORGANIZE directory. At £5 this must be excellent value for all disc users.

CITIZEN PRINTERS

MGT now have the full range of Citizen printers on their catalogue. Sales of the 120D have been very good since they started selling it some months ago and this has prompted MGT to increase the range up to and including a sub £1500 Lazer Printer. MGT have also been appointed by Citizen as an official repair centre.

ZX-MICROFAIR MYSTERY

What is happening to the ZX-MICROFAIR? First rumours were of one in early September, then everyone was talking about the December 10th. Over the last few weeks, however, nobody seemed able to contact the organizers. It now appears that the show is on for the 10th, but its left it a bit late to spread the word. It also means that I may not be able to make it to the show, but I will try.

PLUS D FOR THE +2a

Following the launch by Amstrad of the Plus 2a (or Plus 2 and a half as I call it) many people are coming up against hardware compatability problems. Even simple joystick interfaces don't work. So its good to know that MGT are racing to the rescue. A special version of the PLUS D is being prepared which has a 'Twister' board included. The board changes over various lines to enable the PLUS D (and possibly some other hardware) to work. More news on this next month.

- - - - - -

If you have any news items you want to pass on then send them in. Please mark the envelope NEWS in the top left corner.

CIPHER MACHINE

CKKR JUTK EUR HGBK MUZ SE SKYY6MK TUC EUR IGT HK G YVE

By: Philip Lindon.

CIPHER (sy-fer) a method of altering the letters of a message to conceal its meaning.

CODE a system of word or letters used to represent others, especially in order to keep messages secret.

The first real electronic computers were designed and built, during the last war, to break enemy codes. Computers are ideal for this task as they can plod away, for hours on end (even days if necessary) until the cipher is found.

As an introduction to ciphers and codes (for this article either word will do) I want to look at Caesar's Cipher - named after its inventor, Julius Caesar emperor of Rome. Yes its been around a very long time but its still usable.

Caesar's Cipher works by taking two copies of the alphabet, one normal A-Z and one displaced by a given number of places. For example:-

Normal = ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cipher = EFGHIJKLMNOPQRSTUVWXYZABCD

In the above example the displacement is four. So if we wanted to encode 'FORMAT' we would first look up 'F' in the normal alphabet and then write down the letter from the cipher alphabet thats underneath. In this case that would be 'J', we then continue for the other letters until we get the full word coded as 'JSVQEX'. Quite easy isn't it, but prone to errors if you have a long message so lets get the computer to do the job for us.

Part one of the program inputs a message and then asks for the displacement we want to apply (1 to 25). If you think of it a displacement of 26 would bring us right back to the original alphabet, so 25 is the limit. Now the program looks along I$ (the message) and one by one looks up the characters to go in C$ (the encoded message). Type in part 1 and try it out.

Part 1 of Cipher Machine

```
10 REM SPECTRUM CIPHER MACHINE
20 REM (C)1988 FORMAT
30 REM !! TOP SECRET !!
40 REM FOR THE EYES OF FORMAT
50 REM READERS ONLY.
60 POKE 23658,8
70 LET L$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
80 GOSUB 6000
```

```
 130 INPUT "Your Message";I$
 140 REM TEST FOR VALID STRING
 150 FOR I=1 TO LEN I$: IF NOT (I$(I)=" " OR (I$(I)>="A" AND I
$(I)<="Z")) THEN PRINT ' FLASH 1;"INVALID CHARACTER"; FLASH 0'
';I$( TO I-1); FLASH 1;I$(I); FLASH
 0;I$(I+1 TO ): PAUSE 200: RUN
 160 NEXT I
 170 PRINT ' INVERSE 1;"Message"; INVERSE 0;I$
 180 INPUT "Displacement=";D: IF d<1 OR d>25 THEN  BEEP .5,.5:
 GOTO 180
 190 FOR I=1 TO D: LET L$=L$(2 TO )+L$(1): NEXT I
 200 PRINT ' INVERSE 1;"CODE$"; INVERSE 0;L$
 210 REM **** ENCODE ****
 220 GOSUB 7000
 230 PRINT ' INVERSE 1;"Cipher"; INVERSE 0;C$
 240 STOP
6000 REM HEADING
6010 PAPER 1: INK 7: CLS : BORDER 1
6020 PRINT "*** SPECTRUM CIPHER MACHINE ***"
6030 RETURN
7000 REM CONVERT I$ TO C$
7010 DIM C$(LEN I$)
7020 FOR I=1 TO LEN I$
7030 IF I$(I)=" " THEN  LET C$(I)=" ": GOTO 7050
7040 LET C$(I)=L$(CODE I$(I)-64)
7050 NEXT I
7060 RETURN
```

OK so far? good, but what about decoding messages? Well thats a bit more complicated. Just imagine, having all 25 displaced alphabets laid out before you and then trying each one in turn to work out which gave a readable message. 'RCC JGVTKILD LJVIJ JYFLCU IVRU WFIDRK' would take some time to decode (no I'm not telling you what it is). So lets get the computer to do the hard work for us. Part 2 of our program takes each cipher-alphabet in turn and decodes the input message. It then displays the result on screen together with the displacement used. All you need to do is look at the decoded message and see if it makes sense, if not press enter to see the next try. If you have a very long message to decode just enter the first two lines at first, this should be enough to work out the displacement.

Part 2 - Insert these lines into part 1.

```
  90 PRINT '''"PRESS 1 TO ENCODE MESSAGE"'"       2  TO DECOD
E MESSAGE"
 100 LET R$=INKEY$: IF R$="2" THEN  GOTO 1000
 110 IF R$<>"1" THEN  GOTO 100
 120 GOSUB 6000

1000 REM **** DECODE ****
1010 GOSUB 6000
1020 INPUT "String to Decode =";I$: IF I$="" THEN  GOTO 1020
1030 LET L$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
1040 REM DECODE LOOP
1050 FOR J=1 TO 25
1060 GOSUB 6000
1070 PRINT ' INVERSE 1;I$''
```



```
1080 LET L$=L$(26)+L$( TO 25)
1090 GOSUB 7000
1100 PRINT '' INVERSE 1;"Displacement"; INVERSE 0;J'C$
1110 INPUT "PRESS ENTER FOR NEXT TRY"; LINE R$
1120 NEXT J
1130 STOP
```

So thats the Spectrum Cipher Machine, not bad is it? But it could be just the start for you, if you want get into Ciphers & Codes. Here are a few ideas:-

1) Store the character that is the start of the displaced alphabet at an agreed point in the message so the decoder can pick it up straight away.

2) Every so many characters change the displacement, this should really fox anyone other than the person who has your special decoding routine.

3) Extend the alphabet to include lower case letters and punctuation.

4) Having encoded your message put the result back through for a second pass.

If you produce a good Cipher Machine (or have any contribution to make on Ciphers and Codes) why not send it in to FORMAT. I'm sure I can't be the only one interested in the subject.

+-+-+-+-+-+-+-+-+-+-+

VA#TRACK 6

VA#TRAK 6, A SHARE-PRICE TRACKING AND PORTFOLIO VALUATION SYSTEM

Reviewed By John Wase.

What a remarkable program this is. However, it's not for you if you have only a few hundred BT and British Gas shares and intend to hold on to them - it's for punters with a fair sized portfolio who regularly buy and sell shares. Thus section 17 in the instruction manual on the size and spread of investments lists a range for 10 holdings at a free capital of £10,000 up to 24 holdings at £250,000; section 11 (initial start-up) mentions Liquid Assets £5,000 and Tied Assets at £50,000, although actully, <u>one need not have any money at all</u> if one merely wanted to see how much one could make if one were a capitalist, or, more importantly, the best time to buy and what to buy.

So what does it do? Well, it's based on files, each containing up to 26 weeks of price history on up to 50 different shares or indices. You can look at the list of records, add or delete records, update the file with the latest share prices (takes about twelve minutes for 40 records - should be done each week, but some price interpolation is possible) and value your portfolio. Technical information (for example the "posture" of shares) can be displayed, as can a galaxy of various charts, tables and graphs and these can also be dumped to a printer. You can get profit and loss analyses, and, on a lighter note, even keep track of your own weight whilst you're doing all this.

Once you have a file (and the authors supply a file of the more common equities and indices, updated to the Saturday before dispatch) you will need to buy a copy of the Financial Times each Saturday in order to keep it up to date. Probably the most important thing is to track the posture of shares: postures are carefully described in the manual, for instance a rising posture can arise when the short term moving average price is greater than last week's price, when the medium term moving average is greater than last week's price, or when the short term moving average is greater than the medium moving average. All right, all right; I know this sounds very technical, but the point I am making is that a fair amound of calculation is done by the good old Spectrum - you've really got it working for you - and to get its teeth into it, it needs a fair amount of information, hence the necessity to have 26 weeks prices. Anything older than six months is assumed to have been taken into account already by the market and is therefore not worth bothering about.

Now, the thing to do, if you have sufficient capital, is to buy shares just when they have started to go up, and to sell shares at their peak. And it's worse than that, for you need to select shares which are better-than-average performers in order really to make money. This program helps you with various

calculations to achieve these objectives: hence my comments in the first paragraph.

The program comes on tape along with an updated equities file and a most comprehensive manual of 32 well-filled A4 pages, but rather crudely reproduced from typescript. I found that it was reasonably easy to follow, although my knowledge of the stock market is pretty limited. It is crammed with information, some of it specific to the program, together with a lot of useful general background information of value to the speculator. The rationale behind the program's operations is also given. This consists of a series of philosophies on which are based the algorithms which do the calculations which eventually provide advice on buying and selling.

To help the user, there is a complete section on initial start-up, rather like the tutor file in a Word processor, which guides you in from loading the tape and saving it to disc through many of the features of the program. It can be configured to a variety of rapid storage devices; all you have to do is select "Disciple/+D" on the I/O menu and on the printer port menu and all will be well. The printer itself can be either a "Dinky" one (i.e. Sinclair (any left?) Alphacom or lookalike) or a full size Epson-compatible capable of printing graphics.

In the relatively short time I had the program, I could find no bugs. There is an enormous amount of detailed programming here, done with the greatest care - after all, the authors wrote it for themselves, and being themselves investors, they knew just what was needed. I found it gratifying to see a program of this complexity on the Spectrum: alas this is likely to be almost the last issue (VA#TRAK 6), as they are now converting it to a PC compatible. For those who invest (or want to play at investing), this is a must.

Available from:- Morley Davies Associates, 11 Denham Lane, Chalfont St. Peter, Bucks SL9 0ER. Price £24.75 (upgrades to older versions £18.75)

BACK ISSUES

For members who have missed past issues of FORMAT (or perhaps worn theirs out) we run a back-issue service.

The cost is 70p per issue (90p overseas) incl p&p. Your copies will be sent out as soon as possible but please be patient. Make cheques (drawn on UK bank or Euro-Cheques, P.O., cash) payable to INDUG.

<u>Available Issues</u>

Vol 1 Issues #7 (Feb 1988) - to - #12 (Jul 1988).
Vol 2 Issues #1 (Aug 1988) - to - #3 (Oct 1988).

Please <u>WRITE YOUR ORDER ON A SEPARATE PIECE OF PAPER.</u> DO NOT include letters with order. Remember your membership number or orders will be delayed.

ADDING COMMANDS TO BASIC

By: Nev Young.

Last month we covered the theory of expanding basic on the DISCiPLE or PLUS D. As a practical example I have written a small routine that gives a new COMMAND - "OUT * n,m" - this will delete lines n TO m from a basic program. The * is used to produce a syntax failure in the main ROM. As a number of calls are to the main ROM, a copy of "The Complete Spectrum ROM Disassembly" would be handy for further details of what's going on in the routine.

As you will see the code is in three parts. The first is the autorun code that loads the ONERR variable. The second is the line syntax check, and the third is the code to do the work.

Although it would be possible to to produce code to run on either PLUS D or DISCiPLE, it is not considered a good idea to clutter up the code with such refinements. So remove the three lines that do not apply to your system.

After compiling the code save it to disc by:-

```
SAVE D1"OUT" CODE 65000,74,65000
```

This will cause an auto-run of the machine code when loaded and thus set ONERR to point to the routine.

HOLES TO AVOID.

Commands that do not take any parameters will be executed even if you place parameters after them. For this reason NEW cannot be used as after the NEW completes there is no more statement left.

CLS, as we have seen, can be used, and is by the DOS.

COPY is also used by the DISCiPLE. If you have ever wondered why you cannot use the command COPY SCREEN$ 1 to do a screen dump when in 128K mode the reason is that COPY hangs up waiting for the RS232 printer. In 48K mode it would try to drive a ZX printer but as you can't attach one it drops through to use the DISCiPLE printer.

REM cannot fail syntax so it cannot be used.

RETURN cannot be used as it would do the return from gosub before it failed syntax and so the program sequence would be broken.

CONTINUE can not be used for the same reason as RETURN.

STOP is interpreted differently and can not be used any way. All the other COMMAND words are available (provided the DOS hasn't grabbed them first) as long as the syntax is failed before the last parameter. In my example a * is used to fail the normal OUT syntax.

A trick used with the Interface 1 is to place a * before the COMMAND but because of the way the DOS searches for the start of a statement this will not work. Try * MOVE both with and without the DOS booted.

Finally the snapshot button should NOT be used while your new COMMAND is running or the return flags in the DOS memory will be reset.

One final problem with the DOS that I discovered at the last minute. You cannot use any hook codes in your routine as the DOS assumes that a second error has occurred it then resets the machine stack and returns to the main ROM error handler. This is fixed by lines 460-470, what I do is to change the RST CMR instruction to a CALL. Now the shadow ROM is still paged in when the extension routines are called.

Next month I will be back, with a look at Functions.

```
0010  ; ADD A COMMAND TO BASIC
0020  ;
0030  ; Command - OUT * n,m
0040  ;
0050        ORG  65000
0060  ;
0070  ; MAIN ROM ADDRESSES
0080  ;
0090  NEXT_2 EQU  1C79H
0100  CHADD  EQU  23645
0110  X_PTR  EQU  23647
0120  L_ADD  EQU  196EH
0130  R_CLM  EQU  19E5H
0140  FIND_2 EQU  1E99H
0150  ;
0160  ; DOS ADDRESSES
0170  ;
0180  CMR    EQU  10H
0190  RTBC   EQU  46H
0200  GTNC   EQU  28H
0210  CFSO   EQU  30H
0220  RS18   EQU  2CH
0230  ;
0240  ; DISCiPLE only ADDRESSES
0250  ;
0260  CEOS   EQU  409H
0270  ONERR  EQU  2A6H ;POKE @14
0280  RESP   EQU  0BBH
0290  CADR   EQU  6C7H ;CALL ADDR
0300  ;
0310  ; PLUS D only ADDRESSES
0320  ;
0330  CEOS   EQU  46EH
0340  RESP   EQU  0E7H
0350  ONERR  EQU  200EH
0360  CADR   EQU  210FH
0370  ;
0380  ; Begin by setting ONERR &
0385  ; altering RST CMR
0390  ;
0400  ; page in the DISCiPLE
0410        RST  08H
0420        DEFB 47H
0430  ; load ONERR
0440        LD   HL,start
0450        LD   (ONERR),HL
0460  ; alter rst to call
0470        LD   HL,CADR
0480        LD   (HL),205
0490  ; page out the DOS
0500        OUT  (RESP),a
0510        RET
0520  ;
0530  ; Jump to here if DOS
0540  ; syntax fails
0550  ;
0560  ; test the keyword
0570  start  CP  223  ; jump if
0580        JR  Z,L0 ;  = OUT
0590  ;
0600  ; Else report the original
0605  ;  error
0610        LD   BC,58H
0620        JP   RTBC
0630  ;
0640  ; Get the next Character
0650  L0    RST  GTNC
0660  ;
0670  ; give an error if not = "*"
0680        CP   '*'
0690        JR   NZ,L_ERR
0700  ;
0710  ; get the next two
0720  ; numbers from the basic
0730  ; line. If there aren't two
0740  ; numbers or they are not
0750  ; separated by a comma then
0760  ; Error C will be Called.
0770  ; if all is well the two
0780  ; numbers are placed on the
0785  ; Calculator stack
0790        RST  CMR
0800        DEFW NEXT_2
0810  ;
0820  ; test for the end of the
0830  ; Basic statement
0840  ;
0850        CALL CEOS
0860  ;
0870  ; if we have got this far
0880  ; we must be executing the
0890  ; OUT * Command
0900  ;
0910  ; get Hi line number off
0920  ; Calc stack
0930        RST  CMR
0940        DEFW FIND_2
0950  ;
0960  ; add 1 and save it on
0970  ; the machine stack
0980        INC  BC
0990        PUSH BC
1000  ;
1010  ; get LO line number off
1020  ; Calc stack
1030        RST  CMR
1040        DEFW FIND_2
1050  ;
1060  ; get Hi line # in HL
1070        POP  HL
1080  ;
1090  ; save LO line #
1100        PUSH BC
1110  ;
1120  ; get address of Hi line #
1130        RST  CMR
1140        DEFW L-ADD
1150  ;
1160  ; and save it on m/c stack
1170        EX   (SP),HL
1180  ;
1190  ; get address of LO line #
1200        RST  CMR
1210        DEFW L_ADD
1220        JR   NZ,L2
1230  ;
1240  ; fiddle if line not there
1250        EX   DE,HL
1260  ;
1270  ; get addr of Hi line off
1280  ; m/c stack
1290  L2    POP  HL
1300  ;
1310  ; get the difference
1320        AND  A
1330        SBC  HL,DE
1340  ;
1350  ; Return if LO > HI
1360        RET  C
1370        ADD  HL,DE
1380  ;
1390  ; reclaim memory
1400        RST  CMR
1410        DEFW R_CLM
1420  ;
1430  ; Return ALL DONE
1440        RET
1450  ;
1460  L_ERR
1470  ;  "NONSENSE IN BASIC"
1480        LD   A,0BH ; Error C
1490        LD   (IY+1),A
1500        LD   BC,58H
1510        JP   RTBC
```

WRITING FOR FORMAT

We are always on the look-out for articles and programs to publish in FORMAT.

Articles can be on any subject related to the Spectrum, DISCiPLE, PLUS D or computing in general. From half a page to a long series.

Don't worry too much about spelling and things like that (the Editor can't spell either) we will sort things out. Just put it down as clearly as you can. It is best if you send your article as a word processor file, on disc or tape, but please include a printed copy so we can look at it straight away.

If you want to include any pictures or diagrams then draw them in black, twice the size they need to be in FORMAT. Send them flat, DO NOT CREASE.

If you are sending in a program make sure you give clear instructions. Remember to say what equipment is needed to run the program, i.e. memory required, does it need a disc interface, joysticks, ect. If you can give examples of the output from the program.

Send your work to the address on page 2 or give us a ring to talk about it.

ADVENTURE CORNER

By: Paul Rigby.

Welcome to the latest part of my series for beginners. Last month I talked about locations, movement, the parser and the vocabulary using a working example of a Knight in shining armour who sets out to rescue his Princess. This month I would like to continue with the parser and the vocabulary.

If you remember, we had picked up a golden cabbage from a rather ordinary cabbage patch. If you had the sudden urge to get rid of it you could type "DROP" or "LEAVE" cabbage. However, there may be occasions when you wish to complete an action such as smashing a nearby window. A heavy cabbage such as this would do the job admirably. In which case you might "THROW" or "TOSS" the cabbage at or through the window. Here we come across another facet of adventuring. The role of lateral thinking. The use of objects for relatively unfamiliar situations is a good example of this. Other examples may be to use a football as a float in a river or a pile of books as a step-ladder. So remember to experiment at all times.

Back to our Knight who is patiently waiting on his horse. To the north of him stands the formidable garden shed in which the Princess is locked. So on travelling north to the shed the location description informs you that you can see a shed which stands amongst some cabbages. Again, the player must examine every item within the location. So, an examination of the shed is called for. The reply informs the player that the shed is plain but sturdy with no windows and a strong looking door appears to be the only entrance. Thus, any ideas of throwing the cabbage through a window and rescuing the Princess that way are dashed. After playing an adventure where extensive use is made of the "EXAMINE" command you may notice that it has a definate tree structure whereby an examination of an object leads to discoveries of other objects, an examination of one of those objects may lead to a discovery that it contains several features, and so on.

Each "EXAMINE" leads the player down another branch of the tree. In our example, the examination of the shed has revealed a door. When the door is examined it is found that it is, not surprisingly, locked. Neither pulling, pushing or kicking the door will help. The shed is secure and so is the door. However, how do we know that we have not been carrying a key all along? The command to check this is "I" which stands for Inventory. On typing this command the program displays a list of all the objects that you are carrying. In this case it will be the golden cabbage. Although most adventures allow the use of "I" others require "INV" or "INVEN" or the whole word. So do not be perturbed if a negative response is given when "I" is inputted.

Here we have the classic situation. All the locations have been searched and all of the immediately available objects have been

taken. Let us also suppose that all avenues have been investigated such as examining all of the locations and the objects described within the text, while attempting to perform various actions to no avail. There are occasions, though, when the adventure author has taken pity on the poor adventurer by giving a form of on-line help. This can be accessed by typing "HELP" or "HINT". The solution to the problem may be displayed or a subtle hint may be shown. The majority of games do not have this feature but it is always worth typing the command, if you happen to be stuck on a problem, just in case. The present example has no such feature. At this point many beginners to adventures are tempted to give up. However, it is at <u>that</u> specific point that you should sit down with a cup of tea and think, or take a break and come back later ready to apply a fresh mind to the problem using, you guessed it, lateral thinking!

The answer is to return to the river. Once the river is entered and the horse is treading water again remember that all options for movement are sometimes not given. Here is a case where it may occur. Not through bad design or lack of thought, but actually to enhance the game. Situations such as this are, in my opinion, an acceptable exception to the rule. The direction, of course, is down beneath the surface of the river. With the command "D", or more probably, "SWIM DOWN". Some readers may disagree with me on this point, and if so I would like to hear from you, but I would consider swimming down beneath the waves not an obvious exit to take. Of course this opinion is rather subjective but the player must be aware that there are an awful lot of adventure writers who do not think in the same way as you do. Which is why, when tackling a problem, the obvious solution may not be the correct solution. Other methods should then be tried.

A new location is thus found beneath the river in which a key is found. Simple, all that needs to be done is to "GET" the key, swim to the surface, approach the garden shed and unlock the door. However, adventure authors never let you get away with anything. Always be suspicious when a puzzle looks about to be completed and if possible prepare for disaster. The problem here is twofold. Firstly, time is short because the gallant Knight is holding his breath under water. Secondly, and rather more immediate, he is wearing a suit of shining armour which means that he is too heavy to rise to the surface. The Knight promptly dies.

The player will, in this situation, have to restart the adventure and repeat all of the moves to reach the point just before the Knight dives into the water. Which is fine in this very small example but is extremely tedious in larger adventures. Frequent saving of the position within the adventure means that if your character does die a position close by can be re-loaded. Normally, judicious use of the "SAVE" and "LOAD" commands would be used to save the game onto, and load from, tape. However, the enlightened readers of Format can quickly take a snapshot using PLUS D or DISCiPLE interfaces to save the position to disk. In this, as in all other ways, loading and saving becomes quick and easy instead of a chore.

The armour must be removed, before it is dropped, because it is a piece of clothing. Consequently for any piece of clothing to

take effect it has to be worn after you have picked it up. Meanwhile, the Knight tries again to get the key. In he dives below the waves. The location says that a key lies on the river bed and so, reasonably, the player attempts to "GET" it. Only to be told that;

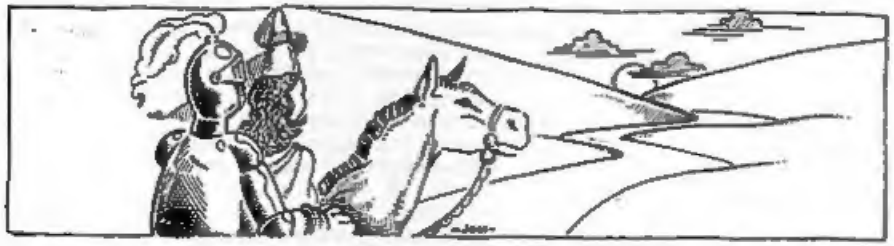
" You cannot carry any more "

Quickly, you drop the cabbage and pick up the key. Whereupon the computer interferes with your progress by informing you that you can hold your breath no longer - you drown. The Red Herring has successfully done its job! Which is what the golden cabbage was all the time. Red Herrings come in all shapes and sizes so the player should be suspicious of every object he or she obtains. Even unusual and valuable items such as golden cabbages. Once again you try to get the key. This time it works! Up to the surface you swim. At this point further deviousness could include the lack of your horse which has swam to the bank and ran away leaving you, a poor swimmer, to be swept away down river and out to sea, or something similar. Again precautions should always be foremost in the adventuring mind. Maybe there is some means of tying the horse's reigns to that root protruding from a dead and floating tree or maybe you could leave him on the river bank, give him some food to keep him occupied and then call him when you surface from the river bed.

When the door of the garden shed is unlocked, by simply saying "UNLOCK DOOR", you can enter the shed by again "ENTER SHED". However, a small minority of adventures may not accept such logicality. Other attempts at entering a building may be successful when saying "GO DOOR" or "GO SHED" or even just "IN" will suffice. Sometimes, I have played the odd game where the latter is the only command accepted!

So to end this excursion for the beginner may I remind you to always try alternative ways of solving a particular problem, alternative uses for everyday objects, alternative directions from those stated in the location description and alternative words for that fussy parser. But above all - examine EVERYTHING!

Next month I will be describing the art of mapping; the only way to prevent yourself becoming lost and to know where everything, such as objects and hazards, is at a glance. But what of the Princess I hear you cry! Well you rescued her; carried her away on your white horse and then you both lived happily ever-after. Well, I'm a sucker for a happy ending.



SAM's SCREEN MODES

By: Bob Brenchley.

SAM is the super, Z80 based, computer that MGT will be launching next year. As I am working very closely with MGT on the project, FORMAT will usually be first with news on SAM. As I am pledged to secrecy I can not answer questions (by letter or phone) regarding the inner workings of SAM. However, after clearing the subject matter of articles with Alan Miles and Bruce Gordon, I will be giving you the full details on SAM over the coming months. Please remember that some fine details are still subject to change.

This month we look at the four screen modes SAM will give you.

MODE 1:- This is the Spectrum compatible mode, the screen is laid out in exactly the same format as the Spectrum. With 8 colours, plus their BRIGHT versions and FLASH, the attribute file works just as you would expect for Spectrum compatability. There are however several bonus points.

First you can set up more than one screen and display the screen of your choice while printing to one of the others. This will give even Basic programmers the opportunity to produce flicker free animation. Next, the 8 screen colours can be selected from SAMs 64 colour palette, so if you don't like the colours in a Spectrum game - just change them.

MODE 2:- A 256x192 pixel screen, with an attribute file that allows you to set the colours on an 8x1 matrix, so you have 8 times the colour resolution as mode 1. Unlike the Spectrum the memory map for the pixels is quite straight forward. Lets say our Mode 2 screen starts in the same place as a standard Spectrum screen. The top left-hand pixel is stored as the first bit of the byte at location 16384 (4000h). The right-hand pixel is the last bit of location 16417. OK thats just like the Spectrum, but if you have studied the Spectrum screen layout you will know that location 16418 contains the pixels for the top row of the character printed at 1,0 - in otherwords 8 pixels down from the first line. I cant go into why the Spectrum does things this way as its beyond the scope of this article, but you can see it in action by doing a FOR-NEXT loop to POKE 255 to locations 16384-22528.

Mode 2 on the otherhand uses a more logical (well to most of us anyway) method. Each line of 32 bytes (32x8=256) is laid out one after the other in memory. So the second row of the character at 0,0 is held at 16416, the third row is at location 16448 and so on. The screen therefore occupies the same space as a Mode 1 screen but the attribute file is 8 time the size. In Mode 2 the attribute file starts (in our example anyway) at

24576 and is the same size as the pixel screen area. This is because each byte in the pixel area (a horizontal line of 8 pixels on the screen) has its own attribute byte. Just add 8192 (2000h) to the pixel byte to get the address of the attribute byte.

The same 8 INK/PAPER colours as mode 1 are available together with their BRIGHT versions and FLASH. Again selected from SAMs palette.

As a mode 2 screen is 12k long it can be held within one of SAMs 16k pages so this mode will be of special interest to games companies who want to pack a lot into the 256k available.

MODE 3:- Now for a mode aimed more at the serious user. Mode 3 is the 80 column mode (well 85.33333 to be exact). The pixel resolution is 512x192 so you have twice the horizontal resolution as other modes. Each pixel can be set to one of 4 colours, selected from SAMs 64 colour palette.

There is no attribute file in this mode. Instead 4 pixels are stored in each byte of screen memory (2 bits per pixel). This means that 24576 bytes are needed to store a screen, but you don't have to cross a page boundry just to access an attribute.

In Mode 3 characters are printed on a 6x8 matrix, hence 85 characters per line. There is no hardware FLASH or BRIGHT, but use of palette switching (I will tell you about that in a future article) will enable you to have flashing characters if you want. Of course with no attributes there can be no colour clash problems so even with only 4 colours I can see some programmers using this mode for games. However it is for word processing, spreadsheets and graph plotting that Mode 3 will be the ideal choice for.

MODE 4:- This is the star attraction, 256x192 pixels each of which can be set to one of 16 colours, selected from the 64 available. Yes pixel resolution colour - NEVER AGAIN will programs have to suffer from attribute colour clash.

The screen is laid out in contiguous memory with each byte representing 2 pixels (4 bits per pixel) hence 16 colours. Again 24576 bytes are required to hold a screen. This is without doubt the mode that most games programmers will flock to use. Because of the way the colour information is stored it will be very easy to give programs fast moving sprites and without clashing problems games should be fast and smooth.

With such a wide choice of modes SAM will attract the best in games and serious software. It will be easy for programmers to convert for SAM (many have alread expressed an interest). If you are a professionl programmer then a detailed development guide will be available in the new year. Give me a ring and it may be possible to add you to the list.

Next month I hope to bring you more details of SAM so keep reading. Remember, if you want to be the first with news of SAM, FORMAT is for you.

CAT SEARCH

By: David Claughton.

This subroutine allows a program to search the disc for a filename returning with variables holding various useful parameters such as the file type and the track and sector on disc where the file starts.

```
 10 CLEAR 29999
 20 DIM s$(1,10)
 30 LET z$="TRACK SECT  T  S  P"
 40 CLS #
 50 LET double density=1
 60 PRINT AT 0,7; INVERSE 1;"INSERT SOURCE DISC"
 70 INPUT AT 21,0;AT 0,0;"Enter Filename: "; LINE f$';"And Driv
e No.: ";drive
 80 LET s$(1)=f$: LET f$=s$(1)
 ■■ PRINT AT 1,10;""";f$;"""
100 PRINT AT 3,1;"FILENAME    ";z$
110 DRAW 0,152: DRAW 255,0: DRAW 0,-152: DRAW -255,0
120 PLOT 0,141: DRAW 255,0: PLOT 0,143: DRAW 255,0
130 LET lnum=5
140 GOSUB 270
150 IF found=0 THEN GOTO 250
160 IF type=0 THEN GOTO 230
170 GOSUB 1000
180 LET lnum = lnum +1: IF lnum <20 THEN GOTO 220
190 PRINT #0;AT 1,0;AT 0,7; INVERSE 1;"PRESS ""M"" FOR MORE"
200 LET i$=INKEY$: IF i$<>"m" AND i$<>"M" THEN GOTO 200
210 INPUT "": LET lnum=5: FOR n=5 TO 20: PRINT AT n,1;TAB 31: N
EXT n
220 IF wild=0 THEN GOTO 250
230 GOSUB 490
240 GOTO 150
250 PRINT #0;AT 1,0;AT 0,10; INVERSE 1;"NO MORE FILES"
260 PAUSE 0: GOTO 10000
270 REM ** SEARCH CATALOGUE **
280 DEF FN c(x)=x-(32 AND (x>=97 AND x<=122))
290 LET wild=0
300 PRINT AT 0,6; INVERSE 1;"SEARCHING DIRECTORY"
310 FOR t=0 TO 3
320 FOR s=1 TO 10
330 LOAD @drive,t,s,30000
340 FOR p=0 TO double density
350 LET headadd=30000+p*256
370 FOR n=1 TO 10
380 LET l1=FN c(PEEK (headadd+n))
390 LET l2=FN c(CODE f$(n))
400 IF l1<>l2 THEN GOTO 470
410 NEXT n
```

```
 420 LET a$="": FOR n=headadd+1 TO headadd+10: LET a$=a$+CHR$ (P
EEK (n)): NEXT n
 430 LET found=1: LET type=PEEK (headadd)
 450 LET track=PEEK (headadd+13): LET sect=PEEK (headadd+14)
 460 RETURN
 470 IF l2=CODE "?" THEN LET wild=1: GOTO 410
 480 IF l2=CODE "*" THEN LET wild=1: GOTO 420
 490 NEXT p: NEXT s: NEXT t
 500 LET found=0
 510 RETURN
1000 REM *** ON FILE FOUND ***
1010 PRINT AT lnum,1; OVER 1;a$;TAB 13;track;TAB 19;sect;TAB 24;
t;TAB 27;s;TAB 30;p
1020 RETURN
9998 REM ** SAVE THIS PROGRAM **
9999 CLEAR : SAVE d1"Search_Cat" LINE 0
```

Save This With GOTO 9998. If you use a single density system change line 50 to : LET double density=0

Now RUN the program. What you should see is a prompt to insert a disc and type in a filename. The program can handle wildcards and letters of the wrong case just as in GDOS. Supply a drive number and you get a display which shows all the filenames which match your input. If there are more than fifteen pressing "M" will scroll through them.

But what use is it? I hear you all saying. Well take a look at lines 20 and 1000-1020. The text stored in Z$ is printed after "FILENAME" on the table and the subroutine at line 1000 is executed everytime a match is found. By altering these it is possible to turn a not very useful program into something a bit more, such as allowing you to have...

HIDDEN FILES

Yes, now you can get rid of all those multi-part programs off your catalogues so that you can see what you've got again!.

The method is actually very simple, in fact I'm amazed Mr Gordon hasn't mentioned it yet. By simply adding 128 to the first byte of the filenames header in the directory the file is no longer listed by the CAT command, yet the file loads, erases etc. as normal. There is as always a catch, however - the system file won't boot if it is hidden, but you can't have everything.

Load in Search_Cat and change line 30 with :-

```
 30 LET Z$="PREVIOUS    IS NOW" (gap is four spaces long).
```

Replace lines 1000-1020 with the following subroutine:-

```
1000 REM *** HIDE FILE ***
1010 PRINT AT lnum,1; OVER 1;a$;TAB 13;
1020 IF type>127 THEN LET type=type-128: PRINT "HIDDEN";TAB 24;"
VISABLE"; GOTO 1040
1030 IF type<128 THEN LET type=type+128: PRINT "VISABLE";TAB 2;"
HIDDEN"
```



```
1040 POKE headadd,type: SAVE @drive,t,s,30000
1050 RETURN
```

Change line 9999 to read :-

```
9999 CLEAR : SAVE d1"Flip_Hide" LINE 20
```

And Save the program with GOTO 9999.

This time on RUNning the program it will flip every matching filename between HIDDEN and VISABLE states depending on what it was previously. Finally , here are some variables you can use while in the subroutine.

```
F$             : holds the input filename.
drive          : holds 1 or 2 for the drive number.
double density : holds 1 for double density or 0 for single.
found          : holds 1 if a match has been found.
wild           : holds 1 if a wildcard is present in f$.
A$             : holds the matched filename.
type           : holds the file type.
track          : holds the start track on disc.
sect           : holds the start sector on disc.
t              : holds the track on the catalogue.
s              : holds the sector on the catalogue.
p              : holds the header no. 0/1 in the sector.
headadd        : holds the address in memory of the header.
```

+=+=+=+=+=+=+=+=+=+

DOS COMMAND CODES

By: Bob Brenchley.

This month I want to explain to you how the User File Information Area (UFIA) works and how you can use it for simple loading and saving.

The UFIA is a 24 byte area of memory used to pass parameters to (or from) the DOS during a Command Code operation. Some of the fields have more than one use but we will go into that later. It looks like this:-

NAME	No BYTES	DESCRIPTION
DSTR1	1	Drive Number (1 or 2)
FSTR1	1	Program Number
SSTR1	1	Stream Number
LSTR1	1	Device Type (d or D)
NSTR1	1	Directory Description (1-11)
NSTR2	10	File Name
HD00	1	File Type (0-3)
HD0B	2	Length of File
HD0D	2	Start Address of File
HD0F	2	Variables Length (Basic)
HD11	2	Autostart Line No (Basic)

The Directory Description byte (NSTR1) is related to the following table:-

1	Basic Program
2	Data Array
3	String Array
4	Code File
5	48k Snapshot
6	Microdrive Type File
7	Screen$ File
8	Special File
9	128k Snapshot
10	OPEN# File
11	Execute File

However the File Type byte (HD00) is limited to 4 different types which are:-

0	Basic
1	Data Array
2	String Array
3	Code File

All files (except OPENTYPE) come under one of these types i.e. HD00 would equel 3 (code) for files with NSTR1 = 4 to 9 and 11 although Special type files (8) could be something else.

Lets work through a practical example, using the UFIA and Command Codes to SAVE an area of memory to disc as a Code file. We will use HOFLE to do the job as this saves a nine byte header which we need for reloading. Here is the assembler listing:-

```
00010   ;GDOS/G+DOS COMMAND-CODE DEMO
00020   ;
00030   ;DEMO #1 - SAVE A CODE FILE
00040   ;
00050           ORG  60000
00060   ;
00070   ;first open file and set up
00080   ;header information.
00090   ;
00100   CSAVE   LD   IX,UFIA ;point to User File Info Area
00110           RST  8  ;call DOS
00120           DEFB 53 ;Command Code - HOFLE 35
00130   ;
00140   ;now save the code block.
00150   ;
00160           LD   DE,(HD0D) ;start address
00170           LD   BC,(HD0B) ;length of file
00180           RST  8  ;call DOS
00190           DEFB 55 ;Command Code - HSVBK 37
00200   ;
00210   ;then close the file
00220   ;
00230           RST  8  ;call DOS
00240           DEFB 56 ;Command Code - CFSM 38
00250   ;
00260           RET  ;return to Basic.
00270   ;
00280   ;the User File Information Area.
00290   ;
00300   UFIA    EQU  $
00310   DSTR1   DEFB 1  ;drive no
00320   FSTR1   DEFB 0  ;not used
00330   SSTR1   DEFB 0  ;not used
00340   LSTR1   DEFB 'd'  ;device type
00350   NSTR1   DEFB 4  ;code file
00360   NSTR2   DEFM 'FILENAME  '  ;10 characters
00370   HD00    DEFB 3  ;file type
00380   HD0B    DEFW 500  ;file length
00390   HD0D    DEFW 30000  ;start addres
00400   HD0F    DEFW 0  ;not used
00410   HD11    DEFW 0  ;not used
```

OK so thats a file saved to disc, now we need to get it back. Well loading a file needs just one more stage, we need to read the header information that is stored in the first nine bytes of a normal file. These are read into the last nine bytes of the UFIA (from HD00) ready for use by the block-read command code. Otherwise its just the reverse of saving the file.

```
00010   ;
00020   ;GDOS/G+DOS COMMAND-CODE DEMO
00030   ;
```

3ACE in +D RAM = CURRENT DRIVE
PAGE IN = IN A,(E7) or RST 8 DB 71
OUT = OUT A,(E7) or JP #BB

```
00040   ;DEMO #2 - LOAD A CODE FILE
00050   ;
00060         ORG  60000
00070   ;
00080   ;first open file and get
00090   ;header information.
00100   ;
00110   CLOAD  LD   IX,UFIA
00120          RST  8
00130          DEFB 59 ; Command Code - HGFLE =3B
00140   ;
00150   ;now read the 9 bytes of
00160   ;header information from the
00170   ;start of the file.
00180   ;
00190          LD   DE,HD00
00200          LD   BC,9
00210   LDBYT  RST  8
00220          DEFB 60 ;Command Code - LBYT  =3C
00230          LD   (DE),A
00250          INC  DE
00260          DJNZ LDBYT
00270   ;
00280   ;next load the block of code
00290   ;
00300          LD   DE,(HD0D)  ;start address
00310          LD   BC,(HD0B)  ;lenth of file
00320          RST  8
00330          DEFB 61  ;Command Code - HLDBK =3D
00340   ;
00350          RET  ;return to Basic.
00360   ;
00370   ;the User File Information Area.
00380   ;
00390   UFIA   EQU  $
00400   DSTR1  DEFB 1  ;drive number
00410   FSTR1  DEFB 0  ;not used
00420   SSTR1  DEFB 0  ;not used
00430   LSTR1  DEFB 'd'  ;device type
00440   NSTR1  DEFB 4  ;code file
00450   NSTR2  DEFM 'FILENAME  '
00460   HD00   DEFB 0
00470   HD0B   DEFW 0
00480   HD0D   DEFW 0
00490   HD0F   DEFW 0
00500   HD11   DEFW 0
```

You will see that the files details are loaded into the 9 bytes starting at HD00 by the code at lines 190-260. This means you dont have to know the start address and its length in order to load a file.

Unlike OPENTYPE files (which we will deal with in a later article) only one file can be open at once.

Well thats all there is room for this month, next time I'll cover error detection and start to look at more complex disc handling. Bye for now.

MORE SPACE SAVING IDEAS MORE

ANOTHER SLICE OF HINTS AND TIPS TO SAVE SPACE IN YOUR PROGRAMS

By: Clyde Bish.

Turning now to on-screen information, graphics etc, if you need a map for a Grid & Compass adventure, or a screen layout for a ladder and platform game, don't waste bytes drawing it from within the program. SAVE the program first so that it autostarts at a line such as - **9999 LOAD "" SCREEN$** - Now design the map or whatever, and SAVE it onto the tape after the main program with - **SAVE "title" SCREEN$.**

If you don't want the display to build up line by line set INK and PAPER to the same colour before LOADing the SCREEN$. This way the picture won't appear until the attributes LOAD in at the end. Don't forget to reset INK after the LOAD or you won't see anything that follows!

Udgs and machine code can also be saved separate from the main program and LOADed in by it. (In fact most assemblers and UDG designers, including the one on the Horizons tape, expect you to do this). SAVE the UDGs with:-

```
SAVE "title" CODE USR "a",168
```

or the machine code with:-

```
SAVE "title" CODE start,length.
```

If you are not using LPRINT in your program you can make the 168 UDG bytes available to the main program by storing the UDG information in the printer buffer. Firstly CLEAR 65535 (This is the last physical byte of RAM). Now you must redirect the system variable UDG to point the start of the printer buffer using - **POKE 23675,0: POKE 23676,91** - then LOAD in the saved UDG bytes to address 23296 within the main program using - **LOAD "" CODE 23296**

You could also store short machine code routine (up to 256 bytes long) there. (as all the UDGs only take up 168 bytes you could store a machine code routine of up to 88 bytes there at the same time!)

It has been mentioned in a previous article but is probably worth repeating here that values of non-changing variables, e.g. those substituting for numbers mentioned last month, can be held in the Variables area only by declaring them on Command mode i.e. with no program lines, as SAVE saves not only the Program area but the Variables area as well. Remember though, RUN must never be used to start such a program as this clears the variable area. Use GOTO line number instead.

If you think all these separate bits of CODE etc will be tedious to LOAD in one after another, fear not. You can do almost all of it in one go. If you include in your program a line such as - **9999 SAVE "title" CODE 16384,49152: GOTO 1** - then the command GOTO 9999 will save all the available RAM from the Display File (with the screen pictures), through the System Variables (plus any

changes you have made), the Printer Buffer (holding any machine code or UDG bytes), the Program area, to the Variables area as CODE so you can LOAD the whole lot in in one go with LOAD "" CODE. When loading is complete the program will autostart at Line 1 (or whatever you include in the SAVE line).

You could also start the Code Save elsewhere in the RAM, if you have no screen display for example. Here are some starting points:-

16384 - saves from the Display file.
23296 - saves from the Printer Buffer.
23552 - saves from the System Variables. (This is the highest starting point).

The length of the bytes to be saved also can be varied:-

```
SAVE "title" CODE 23552,(PEEK 23653+256*PEEK 23654)-23500
```

will save from the Systems Variables to STACKEND. (This is the shortest viable save). If you have machine code bytes in high memory use:-

```
SAVE "title" CODE start,last machine code byte-(start-1)
```

If your machine code needs protecting by a CLEAR make this your first program line (but remember you can't use this if you are trying to hold values in the Variables Area only - they'll be wiped out!).

So to summarise then, suppose you have an adventure which needs instructions, UDG and a short machine code routine, and as much program/variables space as possible, plus an on-screen map, this is how you would set about SAVEing it. First type in a small "driver" program such as:-

```
10 CLEAR 65535: LOAD "" CODE
```

and SAVE it to autostart with:-

```
SAVE "driver" LINE 10
```

Now reset your machine and LOAD in your UDGs and machine code into the Printer Buffer, and the program as normal. POKE the System Variable UDG with 0 and 91 (and make sure your USR calls in the program are to the right address!). Enter the variables in command mode. For example LET o = 0 (with no line number). Finally LOAD in your instructions/title screen and SAVE the whole lot with GOTO wherever you have SAVE "title" CODE 16384,49152: GOTO 1 in your program.

Finally LOAD in the SCREEN$ for the map. (This would be LOADed in by the main program at line 1 in the above example). SAVE this on the tape after the CODE save.

When you reset the machine and type LOAD "driver" this program will LOAD in first, autostart and LOAD in the CODE, the instructions etc appearing on screen first whilst the rest loads in. This program will then autostart and LOAD in the SCREEN$ of the map.

Well, there are the ideas. Now go ahead and byte off as much as your Spectrum can chew! See you again soon.

HACK-ZONE

By: Hugh J. McLenaghan.

This month I will be dealing with a Tape-Header Reader. Without one, your only hope of transferring software is by using the snapshot button, which is not always the best way.

First of all I will give you the listing to type in, then I will describe its uses.

```
 10 REM TAPE-HEADER READER
 20 REM Written By
 30 REM Hugh J. McLenaghan.
 40 REM On 12th OCT 1988.
 50 DEF FN d(x)=PEEK (x+60100)+256*PEEK (x+60101)
 60 CLEAR 59999
 70 RESTORE
 80 LET t=0: LET b=2: LET a=6e4
 90 READ z: IF z<>999 THEN POKE a,z: LET t=t+z*b+(256*b AND z=0
): LET a=a+1: LET b=b+3: GO TO 90
100 IF t<>39631 THEN PRINT "Error in data": STOP
110 DATA 55,62,0,221,33,196,234,17,17,0,205,86,5,201,999
120 CLS
130 PRINT AT 10,10; FLASH 1; INVERSE 1;"Start Tape"
140 PRINT AT 12,8; FLASH 1;"Reading Header"
150 RANDOMIZE USR 6e4
160 LET type=PEEK 60100
170 LET n$="": FOR a=1 TO 10: LET n$=n$+CHR$ PEEK (a+60100): NE
XT a
180 LET bytes1=FN d(11)
190 LET bytes2=FN d(13)
200 LET bytes3=FN d(15)
210 CLS
220 IF type<>0 AND type<>3 THEN PRINT "Type invalid for THIS pr
ogram!": GO TO 330
230 IF type=0 THEN GO TO 280
240 PRINT "Bytes: ";n$
250 PRINT '"Start Address: ";bytes2
260 PRINT '"Length: ";bytes1;" bytes"
270 GO TO 330
280 PRINT "Program: ";n$'
290 IF bytes2>32767 THEN PRINT "No autorun": GO TO 310
300 PRINT "Autorun at Line: ";bytes2
310 PRINT '"Total length: ";bytes1
320 PRINT '"Length without variables: ";bytes3
330 PRINT '''"Press ENTER for a re-run or any"'"other key for co
ntrol."
340 LET a$=INKEY$: IF a$="" THEN GO TO 340
350 IF a$=CHR$ 13 THEN RUN
360 CLS
370 STOP
```

This Header Reader will only give information for BASIC and CODE files. It can be changed for other types, but BASIC and CODE files are used for 99% of all programs. Type in the program and save it by:- SAVE d1"HEAD-READ" LINE 1

If you were to now try it out on a CODE file you might get the following result:-

Bytes: Example1

Start address: 24500

Length: 41036

If you now wish to copy this to disc you would copy down the numbers and name, then reset your computer. Now type:- CLEAR 24499 (start address -1), this will make sure that the CODE will not corrupt the STACK as it loads which would cause the computer to crash.

The next thing you do is to type:- LOAD "" CODE 24500 and start the tape. When loaded you would type:- SAVE d1"name" CODE 24500,41036 to save it to disc. You do not have to save the CODE to disc of course, you could change the CODE and resave it.

Basic can be protected where the code might not be. Programs are now made so that you cannot MERGE them, I will deal with a program which will change this in a later issue. Here are examples of BASIC program headers:-

Program: Example2

Autorun at Line: 10

Total length: 450

Length without variables: 400

Program: Example3

No autorun

Total length: 2000

Length without variables: 2000

Example 1 is a header for a program with a length of 400 bytes, saved to autorun from line 10. It has a variable area of 50 bytes. For this program you can try to MERGE the program. If this fails, then you will have to try another method. Example 2 is a program that does not autorun, has a length of 2000 bytes and has no variables. You could just LOAD this program and then list it.

I am at the moment working on the conversion of OUT-RUN, but it will take a while. This is because a lot of people want this conversion. I will only convert programs that a lot of people want; not just an odd one or two. So if you really want a conversion done, then write me a letter C/O Format and it will be passed on. Any questions will be answered A.S.A.P.

Remember, if you have any pokes, hints or other contributions to make then send them in. Bob has promised me more space if I can fill it with interesting things, but I cant do it all by my self.

Bye for now and see you next month.

MIDI

MIDI FOR BEGINNERS PART 3 - BY RAY ELDER.

This month we look at the way in which MIDI codes are sent and received. Although this does not have a lot of real practical use it does help us to understand what is going on. Most of us will find the programming necessary far too complex, especially as it has to be done in machine code for anything other than sending simple data. Personally, and I suspect for most of us, commercial software will be used.

The MIDI interface uses the UART - Universal Asynchronus Receiver/Transmitter - to transmit and receive data. It uses a serial word format of one start bit, eight data bits, one stop bit and no parity. There are two types of byte, data and control, and these are differentiated by the state of bit 7. If bit 7 is set (1) then it is a control byte and if it is reset (0) then the byte is a data byte.

CONTROL BYTES:- These are usually split into their two component nibbles with the right nibble, bits 0,1,2 & 3, usually indicating the channel number. As this can represent a number 0 - 15 this conveniently can be relatedto the 16 MIDI channels! Remembering that MIDI instruments number the channels from 1 to 16 the number sent or received is equal to one less than the channel i.e. Low nibble = 0010, MIDI channel = 3

1000 Note OFF
1001 Note ON
1010 Polyphonic Key Pressure
1011 Control Change
1100 Program Change
1101 Channel Pressure
1110 System specific messages
(See individual instrument.)
11110111 Code to signify End_of_block

FIG 3a. MIDI Control Codes. Bits 7654

The left or high nibble, bits 4,5,6 & 7, specifies the control action, as bit 7 is always set to indicate that this is a control byte, that leaves us with bits 4,5 & 6 to use for the command. These commands are given in Fig 3a.

DATA BYTES:- These always have bit 7 reset and therefore only the first seven bits are used to send numerical data, this limits us to numbers from 0 - 127. However this is enough to cover all the necessary actions related to MIDI instruments. In note terms this actually covers over ten octaves and there are not many instruments capable of playing that range. Most control bytes are followed by a single data byte, except in the case of note on/off control when two data bytes are expected, these are note pitch and velocity. We will look at these in more detail.

SENDING NOTE DATA:- In all the following I will assume that the

MIDI channel used is number 1, therefore the 'note on' control will be 10010000 or 144 decimal, and 'note off' will be 10000000 or 128 decimal.

The data number for a middle C note is 60 decimal, notes rise or fall by one semitone per number from that. ie. 59=B down from middle C, 61=C sharp up from middle C. Note that very few instruments are capable of playing the full range and most are limited to five or six octaves. Note data outside an instrument's range is either ignored, or the lowest or highest possible note of that name is played.

The velocity byte is supplied for instruments that respond to how fast a key is struck, only the more expensive instruments have this facility and the ones that do not have it simply ignore it. Even if the instrument is not equipped with velocity sensitivity the data must still be sent, usually set to the maximum value - 127. Many instruments do not provide velocity from the keyboard but do react to it via MIDI - the Yamaha FB01 sound module and the RX drum machines do.

Therefore to sound a note of middle C we would send three bytes - 144, 60, 127. This note would sound until either the envelope cut it off as in a piano sound, or until note off information was sent. In the latter case this would be 128, 60, 0. Here there is a curious quirk in the MIDI implimentation. Some instruments do not recognise the 'note off' command and instead you have to sent a 'note on' command with velocity set to zero, ie. 144, 60, 0.

Now before we can send anything there are a few other bits of information that we need to know. These are the addresses of the data receive and transmit registers and the status and control registers. These are given in most of the manuals that are supplied with the interfaces, Fig 3b. gives those for the most common ones. We also need to know which bit of the status byte is used for the transmit/receive accomplished flag, but this only applies when programming more sophisticated programs in machine code.

	MICOM	EMR	SEIL/JMS
Status/Control	63	191	159
Transmit/Receive	191	191	255

FIG 3b. Registers for main interfaces.

```
10 REM send start up message(s)
20 REM replace 'stat' & 'trans'
30 REM with appropriate numbers
40 REM from Fig 3b.
50
60 OUT stat,3:OUT stat,86:OUT trans,
   176:OUT trans,124:OUT trans,176:
   OUT trans,127
70
80 LET A=trans
90 REM send note on data, middle C.
100 OUT A,144:OUT A,60:OUT A,127
110 REM pause/duration of note
120 FOR I=1 TO 1000:NEXT I
130 REM send note off data
140 OUT A,128:OUT A,60:OUT A,0
150 REM This is sent on MIDI channel 1
160 REM Try changing the '60' & 1000
170 REM in the loop (line 120).
```

Program 3a Simple Note On/Off Demo.

THE PROGRAMS:- Program 3a shows how a note can be turned on and off from BASIC. It is worth mentioning that on those Spectrums fitted with a MIDI port you can achieve the same effect by using the PLAY command. Unfortunately there is not a lot of information available on the built in MIDI port. When I contacted Sinclair they were unwilling or unable to give me any details of it. It seems rather limited and not really of much use for anything other than enhancing the built in sound generator.


```
10 REM send start up message(s)
20 REM replace 'stat' & 'trans'
30 REM with appropriate numbers
40 REM from Fig 3b.
50
60 OUT stat,3:OUT stat,86:OUT trans,
   176:OUT trans,124:OUT trans,176:
   OUT trans,127
70
80 LET A=trans
90
100 OUT A,144
105 LET P=40+INT(RND*20)
110 OUT A,P:OUT A,127
120 FOR I=1 TO INT(RND*300+150):NEXT I
130 OUT A,144:OUT A,P:OUT A,0
140 GOTO 100
```

Program 3b. Random Music.

Byte to be sent starts in A, this could be from a table of data, then call this subroutine.

```
        LD B,A
write   IN A,(63)
        RRC A
        RRC A
        JR NC,write
        LD A,B
        OUT (191),A
        RET
```

FIG 3c. Machine code to send a byte.

The byte fetched will be returned from this subroutine in A.

```
read1   IN A,(63)
        AND 1
        JRZ, read2
        XOR A
        IN A,(191)
        RET
read2   IN A,(63)
        AND FOH
        JR NZ,read3
        JR,read1
read3   IN A,(191)
        JR read1
```

FIG 3d. Machine code to get a byte.

Program 3b shows a variation on sending note codes and is a random note player - not very likely to produce a number one hit I'm afraid, but by playing with the frequency of notes and limiting them to the scale then it could probably be made to produce something more tuneful.

To finish for this month, the code in Fig 3c & Fig 3d is an example of how a machine code program could send and receive MIDI data. This is for the Micon (XRI) interface and would have to be modified for any of the other interfaces. In the next issue I will be presenting a program to send program or patch changes to a bank of instruments.

AND FINALLY:- I would welcome any correspondence on MIDI matters, if you have any queries or suggestions (printable) or have produced music of your own I would love to hear from you. I would also welcome some of your hints and tips or stories of your experiences with MIDI. I am still able to offer a tape of music that I have written and recorded using MIDI - This is an audio tape for listening to, not a computer data tape - for the extortionate sum of three pounds & ninty five pence and this may give you some idea of what can be done with a MIDI system.

A MIDI system makes creating music easier, it helps if you have some musical knowledge and ability, but even those with none at all soon find that the due to the ease of editing what you do, you can soon create music which sounds good by the simple means of listening!

Please send all letters and orders for tapes to: Ray Elder, 1 Periton Court, Parkhouse Rd. Minehead, Somerset. TA24 8AE.